



# AI 응용시스템의 이해와 구축

6강. 모델학습 + 에러분석 + 하이퍼파라미터

—  
출석.

# Final Project Discussion

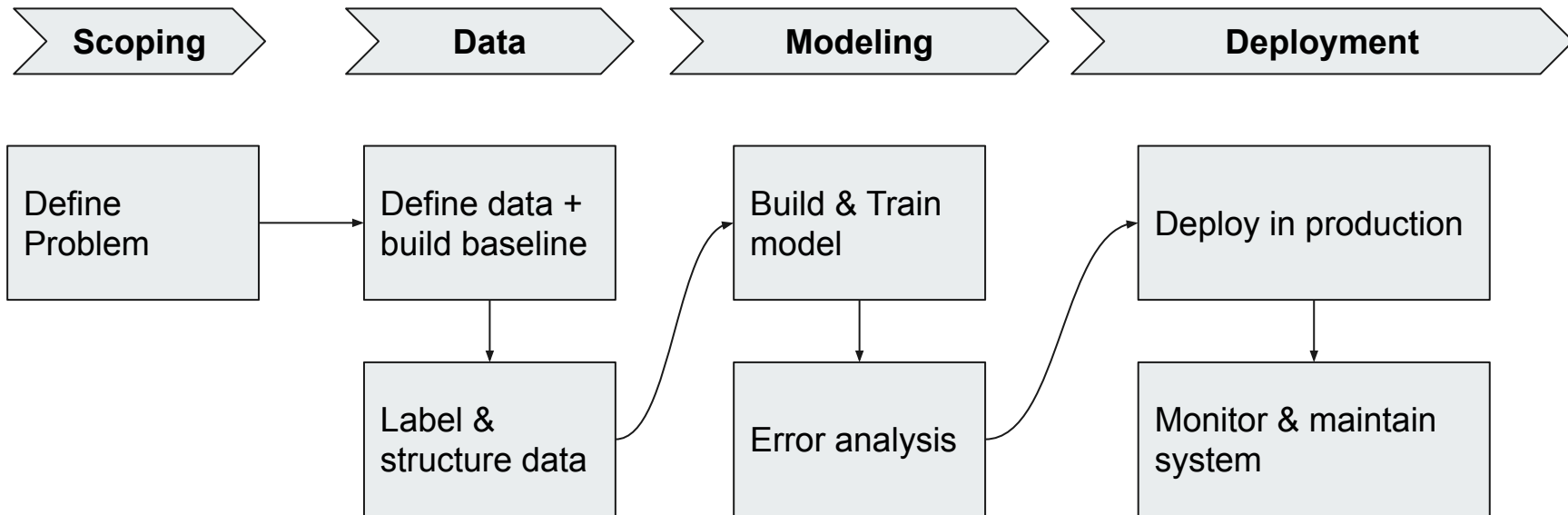


잘 준비 되어 가시나요?

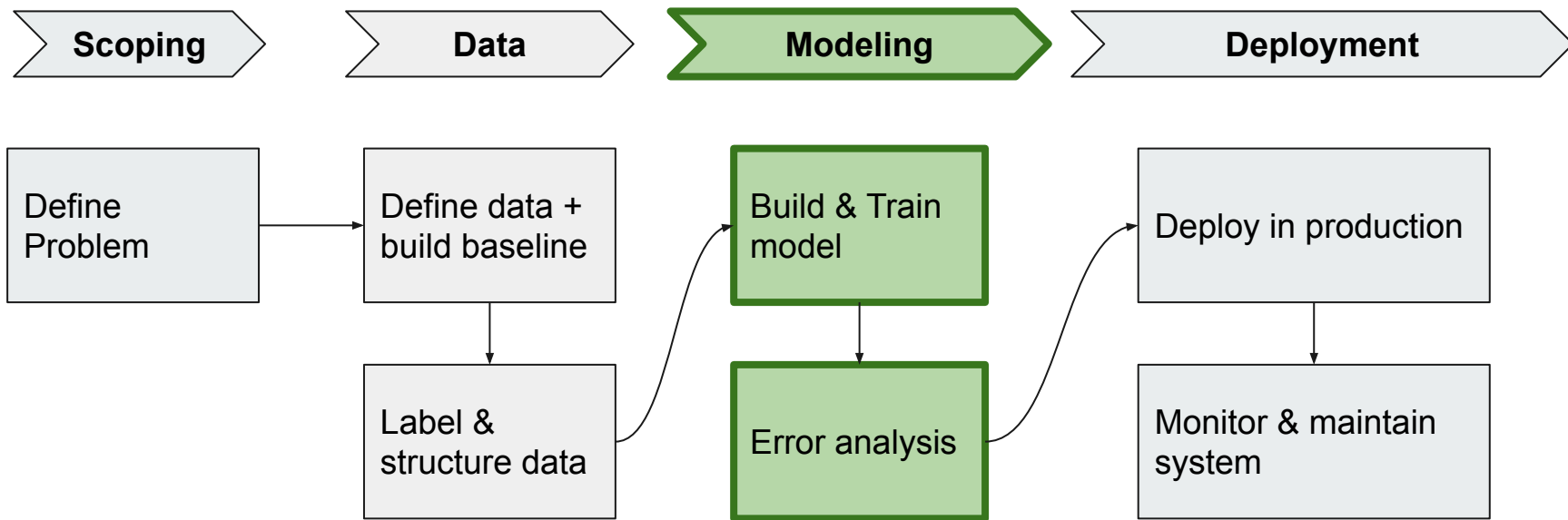
---

# Model Training

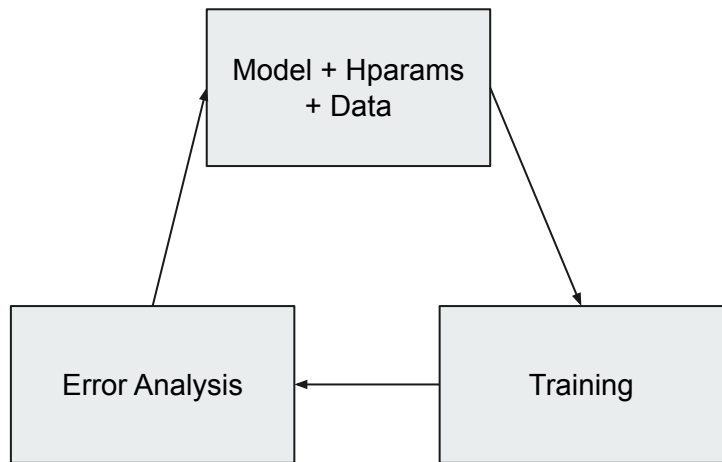
# Overall ML Project Lifecycle



# Overall ML Project Lifecycle

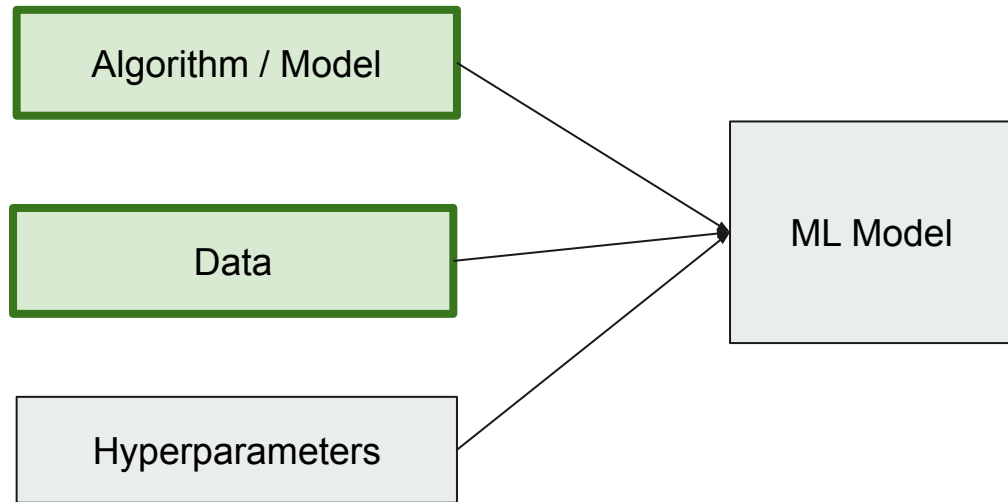
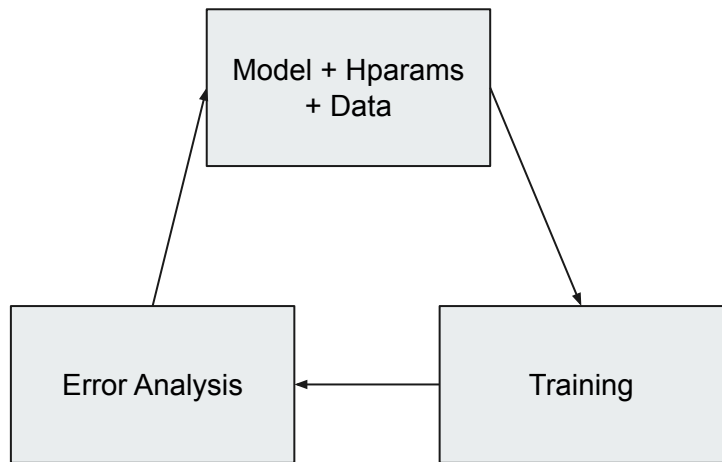


# Modeling as an iterative process



- 여느 소프트웨어 엔지니어링 / 머신러닝 프로세스처럼 늘 단계적인 프로세스로 진행.
- 모델 학습 이후, 에러 분석을 통해 모델성능을 향상.


# Modeling as an iterative process



- 우선 가장 중요한 데이터 + 모델링 알고리즘을 살펴보자.



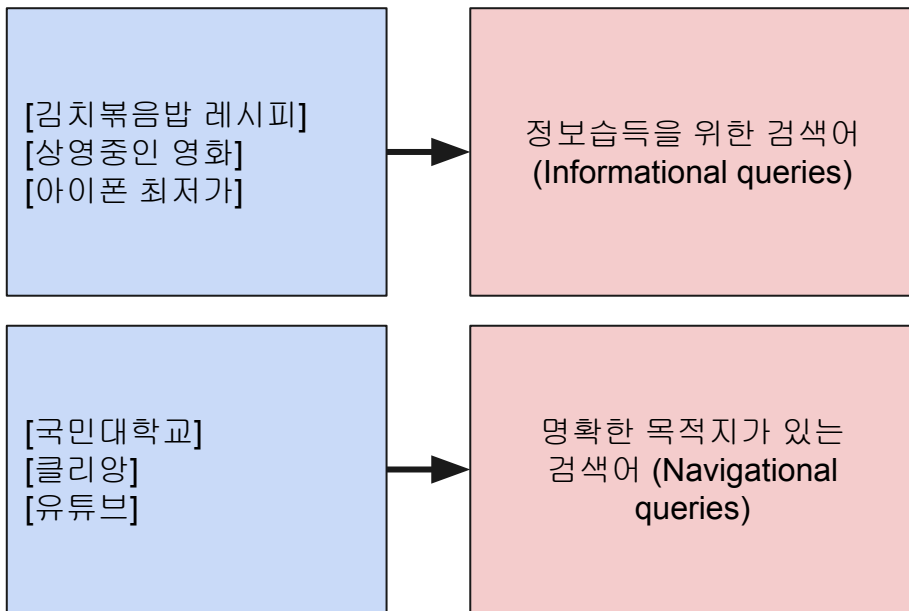
# 모델링이 어려운 이유



1. 학습데이터 (**train set**) 에 대해 좋은 정확도를 달성해야 한다.
2. **dev / test** 데이터셋에 대해 좋은 정확도를 달성.
3. 전체적인 제품의 **퀄리티 (Product-level metric)**를 향상 시켜야 한다.

# Are all inputs equally important?

예) 웹 검색에서 다뤄야 하는 검색어:



informational query보다 중요도가 높음.

- 정확도 (“average accuracy”)는 모든 쿼리를 동일하게 평가
- 중요한 데이터셋을 오버샘플 하거나 가중치 적용 →
  - 늘 도움이 되진 않음.

# 데이터셋 분석

데이터셋 중 중요한 부분 (key slice)이 없는지 분석이 필요함.

## 예1) 신용 대출 심사를 위한 ML 모델

특정 대상의 demographic에게 차별이 가지는 않는지 (인종, 성별, 도시, 언어)

## 예2) 아마존, 쿠팡과 같은 e-Commerce에서의 제품 추천

모든 유저들, 머천트, 제품 카테고리가 공평하게 추천되는지

Model Fairness

중요한 **data slice**에  
해당하는 모델성능의  
지표가 필요.

# Skewed Data set

데이터의 분포도가 어느 한쪽으로 치우쳐 있는 경우 (skewed data distribution)가 종종 있다.

예) 미국 내 환자 분포중, 위암에 걸릴 확률은 매우 낮음.

- 위암이 진단된 환자: 4%, 그렇지 않은 환자 96%.
- (예를들어) 모든 inference에
  - `print(false)` 라고 예측을 한다면?

96% Accuracy

Test data가 accuracy를 사용하여  
모델의 성능을 표현할 수 없음.

---

**Eval using Baseline Data Set**

# Baseline Data Set을 이용한 Evaluation

음성인식 모델중 데이터 슬라이스들에 따른 정확도

Sample Type	Model Accuracy
조용한 방에서의 음성	94%
자동차 소음	89%
주변인 소음	87%
끓기는 노이즈	70%

모델 정확도만 보면  
[끓기는 노이즈]가 있는  
데이터에 대한  
정확도를 더 높여야  
할것 같음.

# Baseline Data Set을 이용한 Evaluation

음성인식 모델중 데이터 슬라이스들에 따른 정확도

3강 슬라이드  
27 참조.

Sample Type	Model Accuracy	Human Performance	Gap from Baseline
조용한 방에서의 음성	94%	95%	-1%
자동차 소음	89%	93%	-4%
주변인 소음	87%	89%	-2%
끊기는 노이즈	70%	70%	0%

Human과 비교했을때, [자동차 소음]이 가장 큰 갭 (headroom)이 있다고 판명.

# Baseline 데이터셋을 어떻게 찾나?

Human Labeling

때로는 Human Labeling을 통해 Irreducible Error를 얻을수도 (3강 슬라이드 32 참조)

연구 페이퍼들을 통한 SOTA 데이터

오픈소스 SOTA 모델들

Heuristic을 사용해서 얻은 데이터

좋은 퀄리티의 데이터 + "괜찮은" 알고리즘  
> 나쁜 퀄리티의 데이터 + SOTA 알고리즘

기존에 사용되는 시스템



---

# 에러 분석 I: 데이터 태그

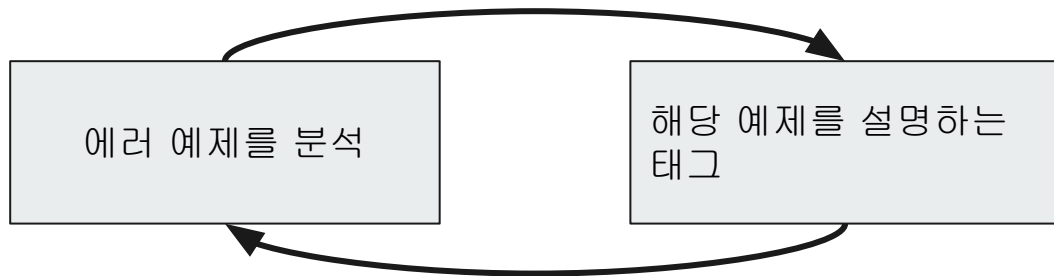
## 에러 분석

모델 학습 후, 예측 에러(prediction error)의 데이터들을 분석하여 어떻게 모델 성능을 향상시킬지 결정한다.

예) 음성인식 모델

에러 예제	Label	Prediction	자동차 소음	주변인 소음	끊기는 노이즈
1	Stir fried lettuce	Stir fry lettuce	Y		
2	Sweetened Coffee	Swedish Coffee		Y	Y
3	Sail away song	Sell away some		Y	
4	Let's catch up	Let's ketchup	Y	Y	Y

## 반복적으로 에러분석 진행



### 환자 질병 진단

- 라벨 클래스: 당뇨병, 패혈증, 고혈압
- 데이터의 특성: 흐릿한 이미지, 여성 데이터, 70세 이상 데이터
- 메타데이터: Xray 기계 종류, 지난 의료진 직종

### E-Commerce 제품 추천

- 사용자 **demographic**
- 상품 **merchant**
- 제품 카테고리

# 에러분석 시, 어떤 지표를 살펴봐야 하나?

전체 데이터중 특정 태그가 몇퍼센트 인가?

적은 양일 경우, Accuracy  
향상에 적은 영향

전체 에러중 특정 태그가 몇 퍼센트인가?

특정 태그를 가진 데이터중, 몇퍼센트가  
잘못 예측되었나?

해당 태그를 가진 예제들의 headroom은  
얼마나 있나?

Human evaluation 대비

이런 지표에 따라 어느 태그의 데이터를 먼저 향상시킬지 결정.

## 어느 데이터를 먼저 향상시킬까?

태그	Accuracy
#조용한곳에서	94%
#자동차소음	89%
#주변인소음	87%
#끊김노이즈	70%

Accuracy만 고려할 경우  
#끊김 노이즈를 우선 향상.

## 어느 데이터를 먼저 향상시킬까?

태그	Accuracy	Human Eval	Gap
#조용한곳에서	94%	95%	1%
#자동차소음	89%	93%	4%
#주변인소음	87%	89%	2%
#끊김도이즈	70%	70%	0%

Human에 대비 갭을  
고려하면  
#자동차소음이  
headroom이 가장 큼.

## 어느 데이터를 먼저 향상시킬까?

태그	Accuracy	Human Eval	Gap	% of data	Accuracy Impact
#조용한곳에서	94%	95%	1%	60%	0.6%
#자동차소음	89%	93%	4%	4%	0.16%
#주변인소음	87%	89%	2%	30%	0.6%
#끊김노이즈	70%	70%	0%	6%	0%

Accuracy Impact를 고려하면  
#조용한곳에서 와 #주변인소음 을 우선  
향상시키는것이 가장 효과적.

$$\text{Accuracy Impact} = \text{Gap} \times \% \text{of Data}$$

## 우선순위를 위한 지표(metric)



추가로, 어느 카테고리의 데이터 + 모델 퍼포먼스를 향상시킬지 정하려면 고려해야 하는 사항들:

헤드룸이 얼마나 있나?

해당 태그 데이터가 얼마나 자주 등장하나?

해당 태그 데이터에 대한 정확도 향상이  
얼마나 어려운지

해당 태그 데이터가 (데이터 빈도와  
상관없이) 얼마나 중요한가?



# 특정 태그 데이터에 대한 모델 향상



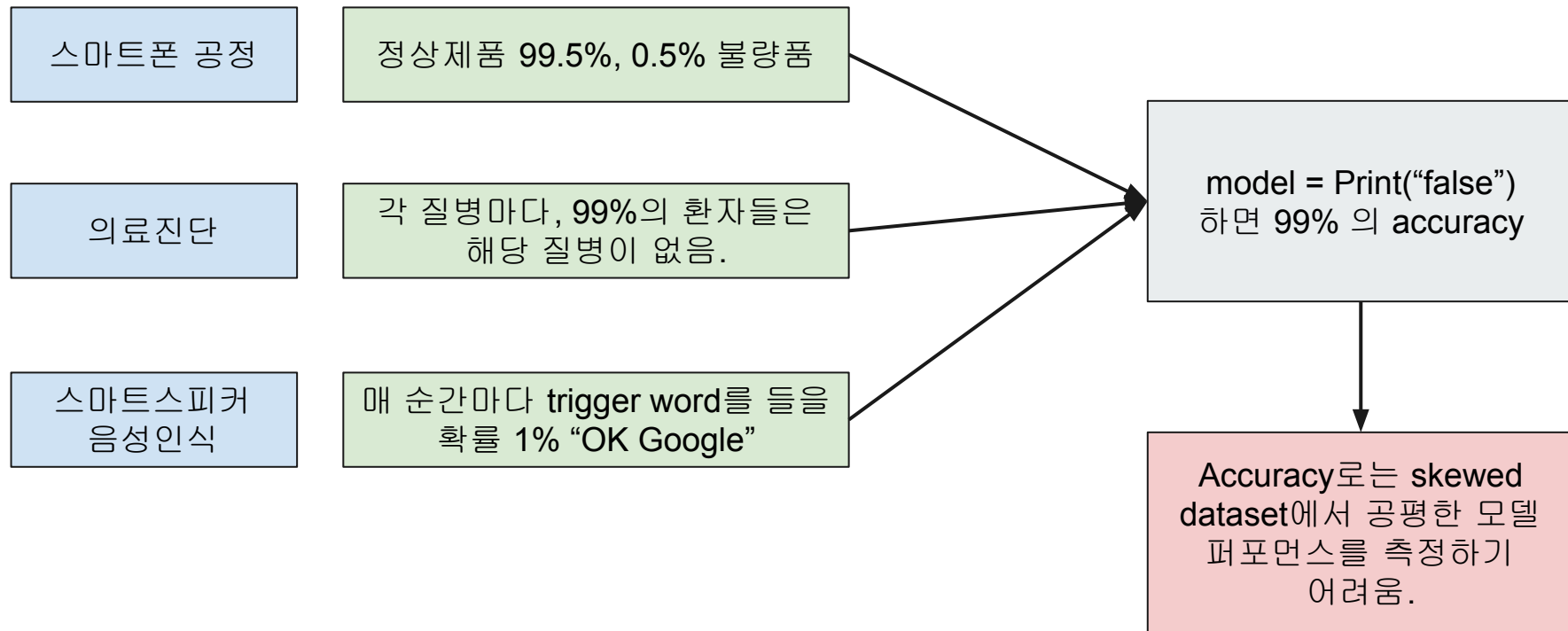
특정 태그 데이터에 대한 모델 정확도를 높이려면:

1. 추가로 해당 데이터를 수집한다
2. 주어진 데이터를 수정하여 늘린다 ([Data augmentation](#)).
  - flip, rotation, scale, crop, translate, noise
  - Heuristic을 사용하여 데이터에 노이즈 형성 (SMOTE)
3. 라벨 정확도 / 데이터 품질을 높인다

---

# 에러 분석 I: Skewed Data

# Skewed Datasets



# Confusion Matrix

	Actual	
	y = 0	y = 1
Predicted y = 0	905 TN	18 FN
Predicted y = 1	9 FP	68 TP

y = 0 쪽으로 skew가 된  
데이터

TN: True Negative  
TP: True Positive  
FN: False Negative  
FP: False Positive

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

$$= 973/1000 = \mathbf{97.3\%}$$

$$\text{Precision} = \frac{TP}{TP + FP}$$

$$= 68/(68+9) = \mathbf{88.3\%}$$

$$\text{Recall} = \frac{TP}{TP + FN}$$

$$= 68 / (68+18) = \mathbf{79.1\%}$$

# Confusion Matrix

		Actual	
		y = 0	y = 1
Predicted	y = 0	914 TN	86 FN
	y = 1	0 FP	0 TP

모든 인풋에 y=0이라고 predict  
한다면?

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

$$= 914/1000 = \mathbf{91.4\%}$$

$$\text{Precision} = \frac{TP}{TP + FP}$$

$$= 0/0 = \mathbf{0\%}$$

$$\text{Recall} = \frac{TP}{TP + FN}$$

$$= 0 / 86 = \mathbf{0\%}$$

y = 0 쪽으로 skew가 된  
데이터

# Accuracy vs Precision + Recall

**Accuracy:** 예측한 모든 prediction 중 올바른 예측을 한 비율.

TN와 TP의 cost가 높을 때 사용.  
(class imbalance가 없을 때)

예) multi class prediction

**Precision:** 예측한 모든 positive 케이스 중, 올바르게 예측한 비율.

FP의 cost가 높을 때 사용

예) 검색랭킹에서 첫번째 페이지의 10개 링크.

**Recall:** 전체 positive 케이스 중, 올바르게 positive라고 예측한 비율

FN의 cost가 높을 때 사용

예) 암환자 식별

# Precision + Recall = F1 score

Precision과 Recall이 서로 tradeoff 가 있는 상황에 어떻게 비교하나?

	Precision	Recall	F1
Model 1	88.3	79.1	83.4%
Model 2	97.0	7.3	13.6%

Real world 데이터에서는 늘 class imbalance가 존재하므로, F1 score가 accuracy보다 종종 더 유용한 지표.

$$F_1 = \frac{2}{\frac{1}{P} + \frac{1}{R}}$$

Precision과 Recall을 조합하여, 둘중 낮은 값을 강조.  
“Harmonic mean” of Precision, Recall.

# 멀티 클래스 예측의 메트릭

멀티 클래스 디텍션: 당뇨, 고혈압, 패혈증, 신부전 -> 모두 흔치 않은 질병이라 accuracy가 올바른 metric이 되줄 수 없음.

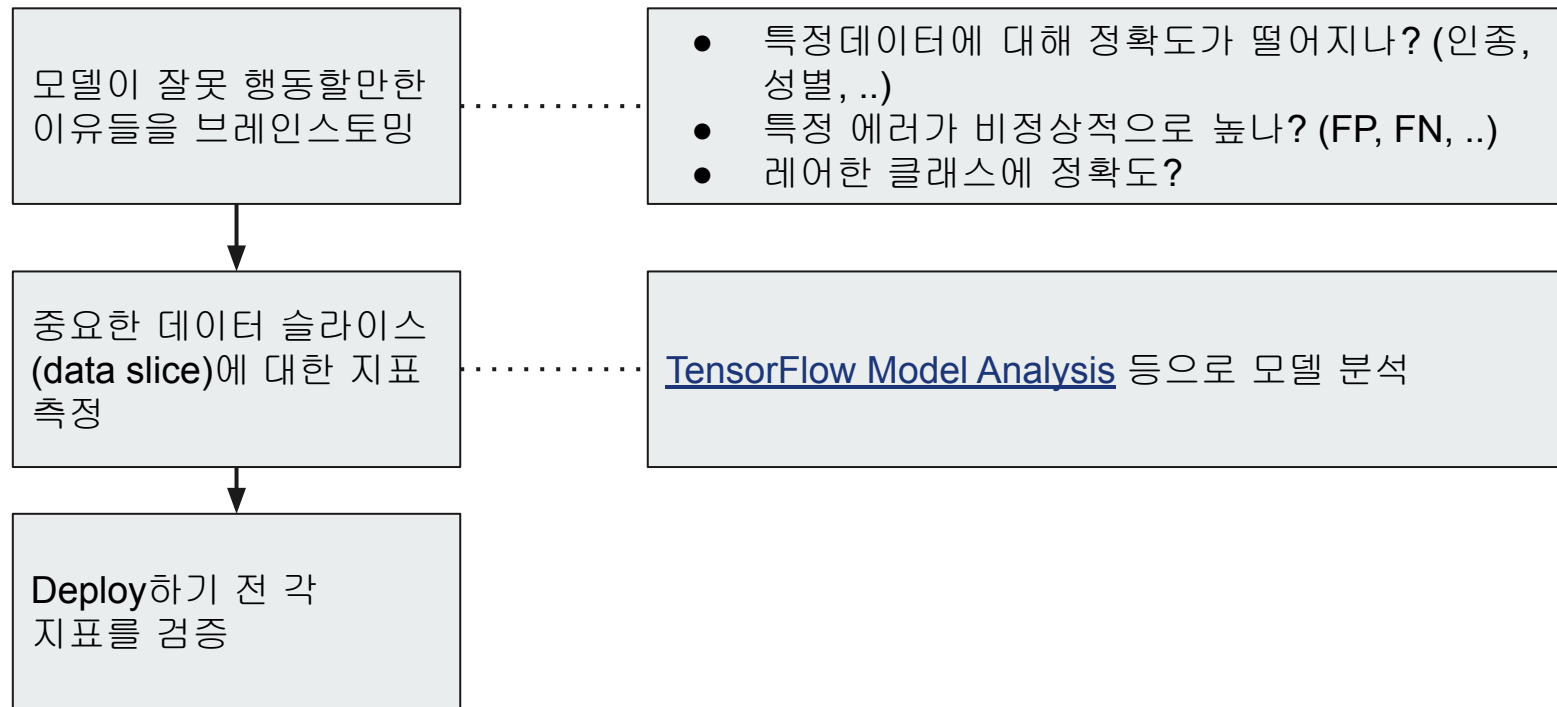
질병	Precision	Recall	F1
당뇨 (diabetes)	82.1%	99.2%	89.8%
고혈압 (htn)	92.1%	99.5%	95.7%
패혈증 (sepsis)	85.3%	98.7%	91.5%
신부전 (AKI)	72.1%	97%	82.7%

참조:

- [F1 score](#)
- [Accuracy vs F1 score](#)



# Deployment Checkup: 프로덕션 직전에 확인할 점들



# Deployment Checkup Example



예: 의료 진단 시스템 (Clinical outcome prediction model)

1. 모델이 잘못 행동할만한 방법들을 브레인스토밍 한다.
  - a. 특정 인종이나 성별에게 잘 예측하는지
  - b. 주중 vs 주말에 다른 결과가 나오진 않는지
  - c. 심각하지만 레어한 질병을 잘 예측 하는지
  
2. 문제될 만한 해당 시나리오에 메트릭을 정의한다
  - a. 각 인종, 성별에 따라 모델 정확도를 따로 측정한다
  - b. 주중 / 주말에 따라 모델 정확도 모니터링
  - c. 중요한 질병들을 향한 모델 정확도를 따로 측정

—

Break

Quiz

---

# Neural Architecture Search (NAS)

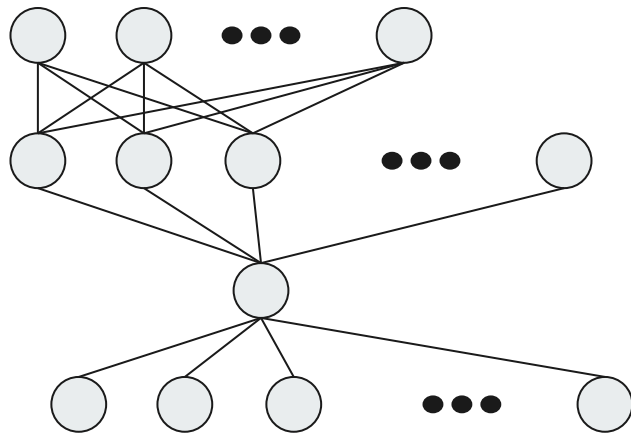
# Refresher: deep neural network

Fashion MNIST 를 사용한 Keras neural network 복습

[Colab](#)

총 4개의 layer로 구성된 모델:

- Flatten layer: 2dim 인풋 feature를 1차원으로 구성
- [Dense layer](#): 512개의 유닛, relu activation
- [Dropout layer](#): 0.2 (regularization 을 통해 overfitting을 방지)
- Dense layer: 10개의 유닛, softmax activation (10개의 multi-class classification)



# Neural architecture search



Neural architecture search (NAS): 뉴럴넷의 디자인을 자동으로 하는 테크닉.

- 최적화된 모델 아키텍처를 찾는 데에 목적을 둔다.
- 일반적으로 검색공간 (search space)가 매우 커다람.
  - 각 hyperparameter마다 자유도
- AutoML: 이 모델 아키텍처를 찾는 데 쓰이는 알고리즘.

예) dense layer의 갯수,  
dense layer당 유닛갯수,  
액티베이션 펄션,...

# ML 모델의 파라미터 종류



## 1. 학습 가능한 (trainable) 파라미터

- a. ML 알고리즘에 의해 학습하는 동안 검색하게 된다.
- b. 예) weights, biases

## 2. 하이퍼 파라미터

- a. 학습 시작하기 전에 정의
- b. 각 training step 이 진행되는 동안 변하지 않는다
- c. 예) learning rate, number of layers, number of units in dense layer

# 하이퍼파라미터 (Hyperparameter)란

모델 및 모델링 파이프라인을 정의할 때 “**학습을 하지 않는 변수**”로서

- 훈련 프로세스
- ML 모델의 토폴로지

를 정의하는 변수이다.

크게 두가지 유형이 존재 함:

1. **Model Hyperparameter:** Hidden layer들의 사이즈나 갯수등 모델 선택을 위한 변수
2. **Algorithm hyperparameter:** SGD(stochastic gradient descent) 의 learning rate나 kNN의 근접 이웃수 등 학습 알고리즘을 정의하기위한 변수



# 하이퍼 파라미터 튜닝



작은 모델이더라도 많은 양의 하이퍼파라미터가 존재할 수 있다.

- 모델 아키텍처
- 액티베이션 평션
- Weight initialization strategy
- Optimization: learning rate, stop condition, ...

매뉴얼하게 튜닝하기엔 어려울 수 있으나 모델 퍼포먼스에 많은 영향!

# Keras Tuner



Keras Tuner:

- TensorFlow 2.0을 위한 하이퍼파라미터 튜닝
- FashionMNIST 데이터셋으로 간단한 모델을 정의하고 Hparam 서치를 실행
  - 두개의 hidden dense layer중 첫번째 layer의 사이즈 (number of units)
  - Adam Optimizer의 학습률 (learning rate)
- **Hyperband**라는 튜닝 알고리즘 사용
  - number of epochs 세팅 가능
- 가장 좋은 hparam검색 후, 그 hparam으로 재학습

# Install Keras Tuner



```
import tensorflow as tf  
from tensorflow import keras  
import IPython
```

```
!pip install -U keras-tuner  
import kerastuner as kt
```

# Get Train data

```
(img_train, label_train), (img_test, label_test) =  
keras.datasets.fashion_mnist.load_data()
```

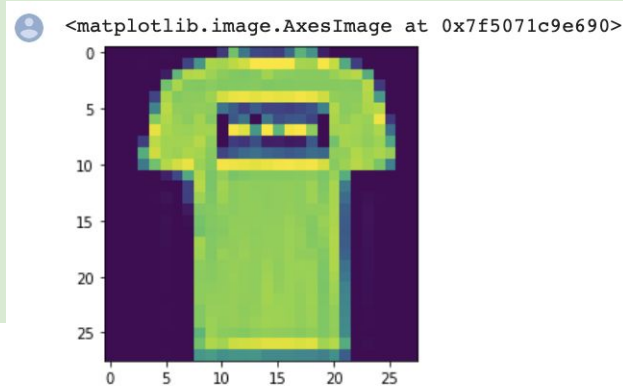
```
# Normalize pixel values between 0 and 1
```

```
img_train = img_train.astype('float32') / 255.0
```

```
img_test = img_test.astype('float32') / 255.0
```

```
from matplotlib import pyplot as plt
```

```
plt.imshow(img_train[1])
```



# Define model with hyperparameter

```
def model_builder(hp):  
    model = keras.Sequential()  
    model.add(keras.layers.Flatten(input_shape=(28, 28)))  
    # Tune the number of units in the first Dense layer  
    # Choose an optimal value between 32-512  
    hp_units = hp.Int('units', min_value = 32, max_value = 512, step = 32)  
    model.add(keras.layers.Dense(units = hp_units, activation = 'relu'))  
    model.add(keras.layers.Dense(10))  
  
    # Tune the learning rate for the optimizer  
    # Choose an optimal value from 0.01, 0.001, or 0.0001  
    hp_learning_rate = hp.Choice('learning_rate', values = [1e-2, 1e-3, 1e-4])  
    model.compile(optimizer =  
        keras.optimizers.Adam(learning_rate = hp_learning_rate),  
        loss = keras.losses.SparseCategoricalCrossentropy(from_logits = True),  
        metrics = ['accuracy'])  
    return model
```

# Define model with hyperparameter



```
tuner = kt.Hyperband(model_builder,  
    objective = 'val_accuracy',  
    max_epochs = 10,  
    factor = 3,  
    directory = 'my_dir',  
    project_name = 'intro_to_kt')
```

# Conduct Hparam Search

```
tuner.search(img_train, label_train, epochs = 10, validation_data = (img_test, label_test), callbacks = [ClearTrainingOutput()])
```

```
# Get the optimal hyperparameters
```

```
best_hps = tuner.get_best_hyperparameters(num_trials = 1)[0]
```

```
print(f"""
```

```
The hyperparameter search is complete. The optimal number of units in the first densely-connected layer is {best_hps.get('units')} and the optimal learning rate for the optimizer is {best_hps.get('learning_rate')}.
```

```
""")
```

```
Trial 30 Complete [00h 01m 28s]  
val_accuracy: 0.8356000185012817
```

```
Best val_accuracy So Far: 0.8452000021934509  
Total elapsed time: 00h 17m 48s  
INFO:tensorflow:Oracle triggered exit
```

```
The hyperparameter search is complete. The optimal number of units in the first densely-connected layer is 352 and the optimal learning rate for the optimizer is 0.0001.
```

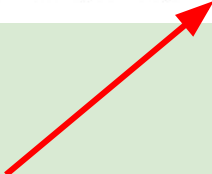
# Define model with hyperparameter

# Build the model with the optimal hyperparameters and train it on the data

```
model = tuner.hypermodel.build(best_hps)
```

```
model.fit(img_train, label_train, epochs = 10, validation_data = (img_test,  
label_test))
```

```
1875/1875 [-----] - 7s 4ms/step - loss: 0.6322 - accuracy: 0.8704 - val_loss: 1.2704 - val_accuracy: 0.8420  
Epoch 9/10  
1875/1875 [=====] - 7s 4ms/step - loss: 0.7297 - accuracy: 0.8741 - val_loss: 1.2189 - val_accuracy: 0.8330  
Epoch 10/10  
1875/1875 [=====] - 7s 4ms/step - loss: 0.6344 - accuracy: 0.8797 - val_loss: 1.0400 - val_accuracy: 0.8498  
<keras.callbacks.History at 0x7f5069a87410>
```





# Keras Tuner



Keras Tuner:

- TensorFlow 2.0을 위한 하이퍼파라미터 튜닝
- [Keras Tuner Colab](#)
  - FashionMNIST
  - Hyperparameters:
    - 첫번째 dense layer의 사이즈 (number of units)
    - 학습률 (learning rate)
  - Hyperband라는 튜닝 알고리즘 사용
    - number of epochs 세팅 가능
  - 가장 좋은 hparam검색 후, 그 hparam으로 재학습 (왜?)
- [Baseline vs Keras Tuned Model Colab](#)
- TFX 에서의 [Tuner component](#)

---

# AutoML

# AutoML



- AutoML의 소개
- Neural Architecture Search
- 검색공간 + 검색 알고리즘
- 퍼포먼스 측정

# AutoML

## Automated Machine Learning (AutoML)

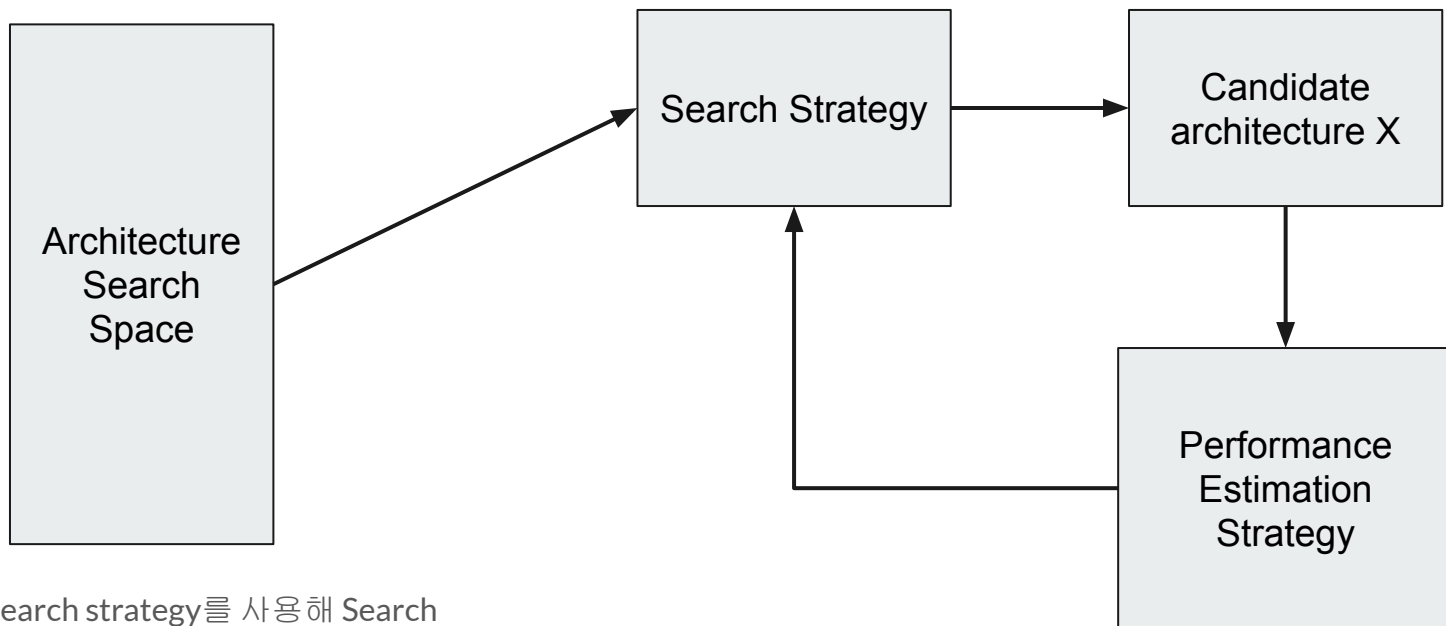
- ML 경험이 풍부하지 않은 개발자도 최적화된 모델을 찾도록 도와주는 테크닉



End 2 End ML 파이프라인 전체를 자동화하는 테크놀로지.

# Neural Architecture Search (NAS)

AutoML 의 한 종류: 모델 아키텍처 선택의 자동화



Search strategy를 사용해 Search space에서 캔디데이트를 선택

# 다음 강의



- NAS / AutoML
  - Search Space
  - Search Strategy
  - Performance Estimator
- Model Optimization Techniques